Erkka Tapola

VIDEOKAMERATEKNOLOGIAN HYÖDYNTÄMINEN ROBOTIN KÄSIAJOSSA

Tieto- ja viestintätekniikan koulutusohjelma
2018

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

VIDEOKAMERATEKNOLOGIAN HYÖDYNTÄMINEN ROBOTIN KÄSIAJOSSA

Tapola, Erkka
Satakunnan ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutusohjelma
Huhtikuu 2018
Sivumäärä: 31

Asiasanat: ohjelmistotuotanto, videotekniikka, Java, avoin lähdekoodi

---

Tässä opinnäytetyössä pyrittiin täyttämään Cimcorp Oy:n tarve saada toistettua kameroilta haettua videokuvaa robottisolun käyttöliittymällä. Cimcorp Oy:n robotteja ohjataan välimatkan päästä käyttöliittymältä, ja ajoittain näköyhteyttä ohjattavaan kohteeseen ei ole. Tällaisissa tilanteissa robotilta haettu ja käyttöliittymällä esitetty videokuva on erittäin tärkeää robotin ohjattavuuden kannalta. Käyttöliittymällä toistetusta videokuvasta on mahdollista saada myös muita hyötyjä, joita tässä työssä tutkittiin.

Tehty ohjelmisto suunniteltiin ja toteutettiin yhdessä Cimcorp Oy:n tuotekehityksen ja robottisolu-tuotteesta vastaavien henkilöiden kanssa. Ohjelmiston kehityksessä käytettiin apuna avoimen lähdekoodin OpenCV-kirjastoa ja Jenkins-, Git- ja Gradle-työkaluja. Tämän myötä valmistunut ohjelmisto siirtyi Cimcorp Oy:n tuotekehityksen käyttöön, jotta he voivat jatkossa käyttää ja kehittää sitä haluamallaan tavalla.

Lopputuloksena syntyi Javalla koodattu CcIpCam.jar-kirjasto, joka hakee lähetetyn videokuvan ja toistaa sen JLabel-elementissä. CcIpCam.jar:a saatiin työssä onnistuneesti käytettyä Cimcorp Oy:n tuotteissa, ja tämän seurauksena luotu ohjelmisto on tarkoitus ottaa käyttöön Cimcorp Oy:n tuotannossa mahdollisimman pian. CcIpCam.jar on hyvin yleiskäyttöinen, sillä se on mahdollista integroida käytännössä mihin tahansa Java-pohjaiseen käyttöliittymään.

USE OF A VIDEO IMAGE WHEN DRIVING ROBOT IN MANUAL MODE

Tapola, Erkka
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information and Communication Technology
April 2018
Number of pages: 31

Keywords: software engineering, video technology, Java, open source

---

The aim of this study was to fulfill the demand of Cimcorp Oy to display fetched video image on a user interface of a cell. Robots manufactured by Cimcorp Oy are controlled from a distance by a user interface and occasionally there is no visual contact to the target. In this kind of situations the video image from the robot is vital in order to manually control driving of the robot. It is also possible to gain other benefits from the displayed video image and these benefits were investigated in this study.

Produced software was defined and designed together with product development and personnel in charge of the cell in Cimcorp Oy. Open source library OpenCV was used in the development of this software and tools such Jenkins, Git and Gradle were big part of the development. After finishing with the software development the software was handed to Cimcorp Oy for use of the software in production and for further development.

The result of this study was a Java-based library CcIpCam.jar that fetches the sent video image and then displays it in JLabel-element. During this study CcIpCam.jar was successfully used in the products of Cimcorp Oy and therefore the created software is planned to be put in production as soon as possible.

SISÄLLYS

1 JOHDANTO

Tämä työ on tehty täyttämään Cimcorp Oy:n tarve löytää ratkaisu robotin ajamiseen käsiajolla sellaisissa tilanteissa, jossa robotin käyttöliittymältä ei ole suoraa näköyhteyttä kohteeseen, johon tai jossa robottia halutaan ohjata. Työssä on myös selvitetty muita mahdollisia käyttökohteita robotilta haetulle videokuvalle, tutkittu tallennetusta videokuvasta saatuja etuja, sekä listattu erilaisia haasteita videokuvan tallentamiseen liittyen.

Pääsääntöisesti Cimcorpin robotit ovat portaalirobotteja (kuva 1), jotka on suunniteltu nostamaan joko yksittäinen tuote tai pino tuotteita lattialta ylös ja siirtämään toiseen positioon. Tuotteiden käsittelyä varten portaalirobottien tarttujat varustetaan web-kameralla. Usein muut lattialla olevat pinot saattavat aiheuttaa näköesteitä robottia etäältä ohjattaessa, joten videokuvan saaminen robotilta käyttöliittymälle on monissa käsiohjaustilanteissa välttämätöntä.

Käyttötarkoituksen käsittelyn jälkeen työssä esitellään siinä käytetyt komponentit, jotka on hankittu sekä Cimcorpin emoyhtiöltä Murata Machinerylta, että kameravalmistaja Axisilta. Tämän jälkeen paneudutaan itse ohjelmiston rakenteeseen ja sen kehittämiseen, sekä sen kehityksessä törmättyihin ongelmiin ja haasteisiin.

Tämän jälkeen työssä käsitellään siinä käytetty testausympäristö hyvine ja huonoine puolineen, ja lopuksi paneudutaan myöskin itse kuvanhaun toteutuksessa käyttäjärajapinnalle käytettyihin työkaluihin ja apuvälineisiin, eli Gitiin, Gradleen ja Jenkinsiin.

2 KÄYTTÖTARKOITUS

2.1 Videokuva käsiajossa

Suurin tarve Cimcorp Oy:lle saada varustettua robotti videokameralla oli helpottamaan robotin käsiajoa tilanteissa, joissa näköyhteys robotin käyttöpäätteeltä ajettavaan kohteeseen on huono. Huonoa näköyhteyttä käsiajon aikana saattavat aiheuttaa varastossa olevat tuotteet, tilan huono valaistus, päätteen sijoituspaikka tehtaassa tai varastoalueen suuri etäisyys käyttäjästä.

Videokuva robotin käyttöliittymässä helpottaa käyttäjän käsiajoa monissa tilanteissa. Ilman videokuvaa esimerkiksi yksittäisen tuotteen nostaminen lattialta muiden pinojen takaa käyttäen robotin manuaaliajoa ja ilman suoraa näköyhteyttä on käytännössä mahdotonta. Robotin joutuessa virhetilaan esimerkiksi virheellisen poiminnan takia ei käyttäjällä ilman kameraa olevalla robotilla välttämättä ole tarkkaa tietoa siitä, mikä tilanne robotin poimintapaikalla on.

Turva-aitojen ja muiden fyysisten rajoitteiden takia robotin lähelle pääseminen ja tilanteen läheltä näkeminen ei aina onnistu, vaan tilannetta joutuu arvioimaan kauempaa. Käyttöliittymällä olevan kamerakuvan ansiosta tällaisien virhetilanteiden selvittäminen ja niistä toipuminenkin on helpompaa, kun tilannetta voi seurata kamerakuvan perusteella hyvinkin läheltä.

Kuva 1. Cimcorpin portaalirobotti

2.2 Videokuva automaattiajossa

Käyttöliittymällä on mahdollista nähdä videokuvaa robotin toiminnasta myös robotin ollessa automaattiajossa. Kuvaa katsomalla pystytään tarkasti seuraamaan, toimiiko robotti halutulla tavalla ja ongelmatilanteisiin törmättäessä todistamaan, mistä aiheutunut vika johtui.

Robotin ollessa automaattiajolla on kamerakuvalta helppo seurata, mikäli robotti toimii jollain tapaa ei-halutusti ja tässä tilanteessa robotti voidaan pysäyttää ennen kuin mitään vakavaa ehtii sattua.

2.3 Tallennettu videokuva

Optiona laajentaa tätä työtä jälkeenpäin tehtiin esitutkimus robotilta haetun videokuvan nauhoituksen toteuttamisesta. Videokuvan nauhoitus sisällytettiin työhön optiona, sillä sen toteuttamiseen liittyy haasteita, jotka laajentavat työtä entisestään. Tästä syystä videokuvan tallennuksen toteutus on asia, joka voidaan toteuttaa jatkokehityksen yhteydessä.

2.3.1 Tallennetun videokuvan edut

Tallennetusta videokuvasta on mahdollista saada monia hyötyjä, jotka sekä helpottavat operaattorien toimintaa että auttavat virhetilanteiden todentamisessa ja ratkomisessa jälkikäteen. Tallennettu videokuva olisi suurena apuna robotille jo aikaisemmin tapahtuneiden virheiden selvittämisessä: Mikäli robotti on esimerkiksi yön aikana ajautunut virhetilaan tai virhetilanteeseen päätyminen on muusta syystä jäänyt operaattorilta näkemättä, pystytään jälkeenpäin käymään läpi virheen aiheuttanutta tilannetta videolta ja yksityiskohtaisesti selvittämään, mistä vikatilanne johtui. Näin videotallenteesta saataisiin kiistaton hyöty ongelmia ratkoessa jälkikäteen, eikä ongelmia tarvitsi välttämättä aina havaita reaaliajassa silmällä. Tämän ansiosta operaattorin ei tarvitsisi seurata vierestä ja odottaa virheen tapahtuvan, vaan virheen syyt ja ajautuminen virheeseen voitaisiin jälkikäteen todentaa nauhoitteelta.

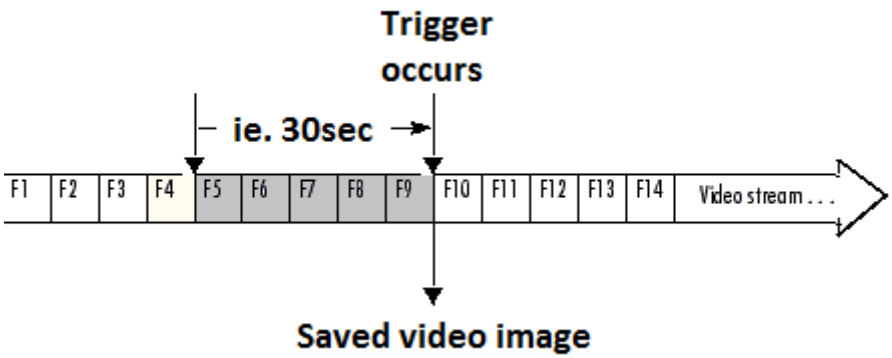
Nauhoitettuja videomateriaaleja voisi käyttää myös koulutusmateriaaleina. Koulutustilanteissa materiaaleja olisi mahdollista käydä läpi ja selvittää esimerkiksi operaattorin toimintaa erilaisissa virhetilanteessa. Tällä pyrittäisiin varmistamaan operaattorin oikeaoppinen toiminta kussakin tilanteessa. Ilman videomateriaaleja kaikkia virhetilanteita voi olla vaikea simuloida ja käydä läpi, mutta mikäli ne kaikki olisi koottu nauhoitteille, olisi niiden läpikäynti helppoa.

Videomateriaaleja olisi mahdollista käyttää myös apuna ohjelmistokehityksessä. Mikäli robotin ohjelmaan tehtäisiin muutos ja sen toimintaa haluttaisiin verrata vanhaan toiminnallisuuteen, olisi tämä helppoa tallennetun videomateriaalin avulla. Kamerakuvasta voitaisiin ensin seurata robotin toimintaa ja käyttäytymistä uuden ohjelmiston kanssa ja sen jälkeen verrata sitä nauhoitteessa näkyvään toimintaan ennen muutosta. Näin nähtäisiin uuden päivityksen tuomat erot vanhaan.

2.3.2 Videokuvan tallentamisen haasteellisuus

Saadun videokuvan tallennus osoittautui osaltaan ongelmalliseksi. Mikäli videokuvaa tallennettaisiin jatkuvasti kaikista robotin liikkeistä, tulisi tallennettavaa materiaalia todella paljon. Tässä tilanteessa vain pieni osa tallennetusta materiaalista olisi hyödyllistä ja tämän hyödyllisen materiaalin löytäminen kaiken muun joukosta olisi työlästä.

Pyrkiäksemme välttämään ylimääräisen tallentamisen, tarvitsisi tallennustoimintoon kehittää ns. liipaisu-periaate (kuva 2). Videokuvaa tallennettaisiin jatkuvasti, mutta esimerkiksi yli 30 sekuntia vanhat videot poistettaisiin automaattisesti. Järjestelmän joutuessa virhetilaan se antaisi ohjelmalle signaalin, jonka perusteella ohjelma jättäisikin poistamatta sitä edeltäneen videotallenteen. Näin saataisiin virhetilanteesta nauhoite talteen ja samalla karsittua ylimääräinen videomateriaali.



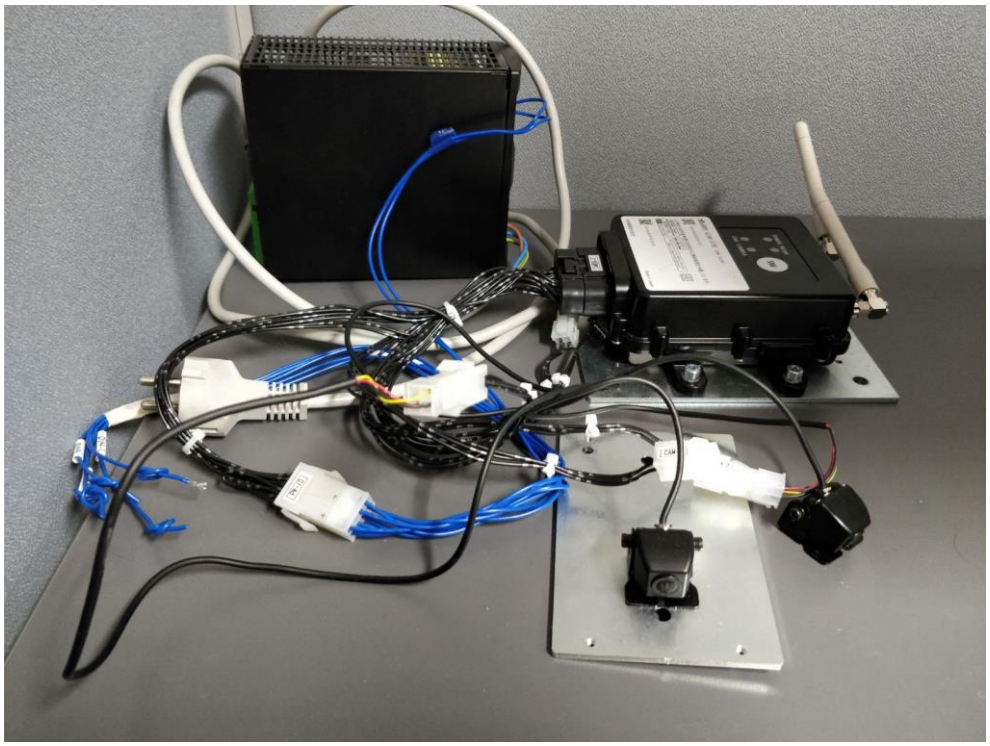
Kuva 2. Videokuvan tallentaminen liipaisu-periaatteella

Lisäksi videoita tallentaessa ongelmaksi muodostuu niiden muistilta vaatima tila. Videotiedostot ovat videon laadusta riippuen usein melko suuria ja pitkässä ajanjuoksussa niitä kertyy paljon. Tämän takia videoita ei olisi mahdollista tallentaa lokaalisti robottia ohjaavalle soluohjaimelle, sillä sen muistin määrä on hyvin rajallinen, vaan videoiden tallentaminen olisi tapahduttava verkon yli toiseen kohteeseen.

3 TOTEUTUS

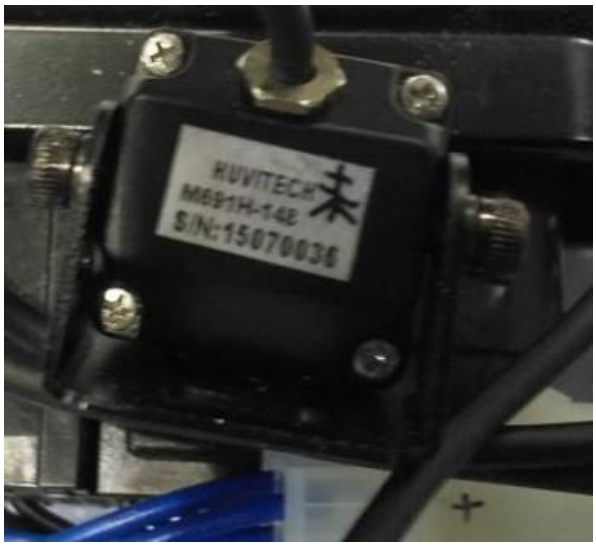
3.1 Komponentit

Cimcorp Oy on Murata Machineryn tytäryhtiö. Murata Machinery on omissa tuotteissaan käyttänyt itse valmistamiaan komponentti-kokonaisuuksia (kuva 3). Muratan ja Cimcorpin tiiviin yhteistyön ansiosta oli luonnollista hankkia työssä käytettävät laitteet sieltä. Murata oli aiemmin jo todennut kyseisen komponenttiratkaisun toimivaksi, joten oli helppo päätös seurata jo hyväksi todettua linjaa komponenttien valinnassa.



Kuva 3. Murata Machinerylta tilatut kamerat, wlan-lähetin ja virtalähde

Erityisen hyvää Muratan toimittamissa komponenteissa oli kameroiden pieni koko (kuva 4). Pienen kokonsa ansiosta kamerat eivät haittaa robotin liikettä, joten niiden kiinnityskohdille oli monta eri mahdollisuutta. Kameroista saatu videokuva ei ole hd-laatuista ja linssin pyöreän rakenteen takia kuva on kokenut pientä tynnyrivääristymää, mutta kuvatarkkuus on kuitenkin riittävä robottien ohjaamiseen, eikä kuvan vääristymäkään haittaa ohjattavuutta.

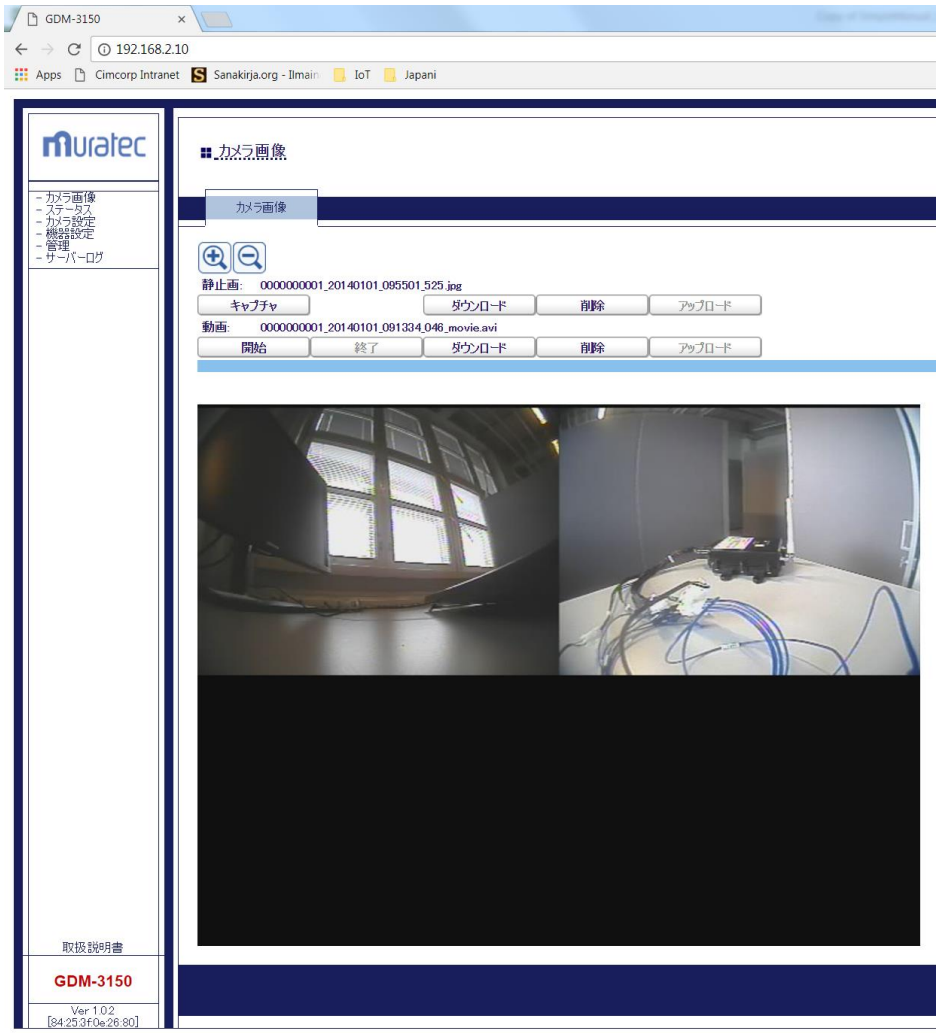


Kuva 4. Kamera

Kamerat on kytketty johdolla wlan-lähettimeen, joka lähettää kameroiden kuvan ylläpitämälleen käyttöliittymälle (kuva 5). Käyttöliittymältä on mahdollista esimerkiksi tarkastella kameroiden lähettämää kuvaa, valita kuinka monen kameran kuvaa halutaan näyttää tai ladata kameran lähettämää kuvaa tiedostoksi tietokoneelle. Kameroita on mahdollista kytkeä kerrallaan korkeintaan neljä per wlan-lähetin.

Kuva 5. Kameran kytkentä soluohjaimeen

Kameroiden pienen koon lisäksi hyvää Muratalta tilatuissa komponenteissa oli wlan-lähettimen web-käyttöliittymä ja toiminnallisuus. Koska Murata oli jo omissa tuotteissaan käyttänyt kyseisiä komponentteja samaan tarkoitukseen, oli wlan-lähettimen ohjelmisto suunniteltu juuri tätä käyttötarkoitusta varten. Se sisälsi valintamahdollisuuden aktiivisten kamerakuvien lähettämistä varten, kuvien kääntämisen ja jopa tallentamismahdollisuuden (kuva 6). Hyvä ohjelmisto ja yksinkertainen käyttöliittymä mahdollistivat sen, että kameroilta saadun kuvan toistaminen web-palvelimella onnistui kätevästi.



Kuva 6. Kuva wlan-lähettimen web-käyttöliittymästä

Murata Machinerylta tilatuissa komponenteissa on tuotantotarkoitukseen nähden kuitenkin ongelma, sillä niitä ei ole EU-hyväksytty. Tästä johtuen mahdollisuutta vaihtoehtoisten toimittajien komponenttien käyttämisessä Cimcorpin tuotannossa jouduttiin myös selvittämään. Ongelmaksi kuitenkin muodostui, ettei vastaavanlaisia komponentti-paketteja juurikaan ole markkinoilla. Tästä johtuen Cimcorpilla onkin nyt selvityksen alla, mitä kaikkea EU-hyväksynnän saaminen Murata Machineryn komponenteille vaatisi.

3.2 Vaihtoehtoiset komponentit

Murata Machinerylta tilattujen komponenttien lisäksi ohjelmistoa haluttiin testata myös muilla komponenteilla. Syitä tähän oli mm. CE-hyväksynnän puuttuminen Muratalta tilatuista komponenteista, kyseisten komponenttien mahdollisesti epävarma saatavuus jatkossa ja koodatun ohjelmiston soveltuvuuden testaaminen muiden komponenttien kanssa. Vastaavanlaisia komponenttikokonaisuuksia oli kuitenkin saatavilla hyvin vähän, eikä täysin samanlaista pakettia testaamiseen löydetty lopultakaan.

Vaihtoehtoisiksi komponenteiksi lopulta valikoituivat Axisin valmistama P1214-E-kamera ja PhoeniXin valmistama wlan-lähetin (kuva 7.). Tämä yhdistelmä ei kuitenkaan salli kuin yhden kameran kuvan näyttämisen kerrallaan. Jotta käyttäjälle saataisiin näytettyä monipuolista kuvaa robotin liikkeistä kahdesta suunnasta ja kahdesta kamerasta, joitain muutoksia kyseinen komponenttikokonaisuuskin vaatisi tulevaisuudessa. Kaksikameraisia kokonaisuuksia on Axisiltakin kuitenkin saatavilla.

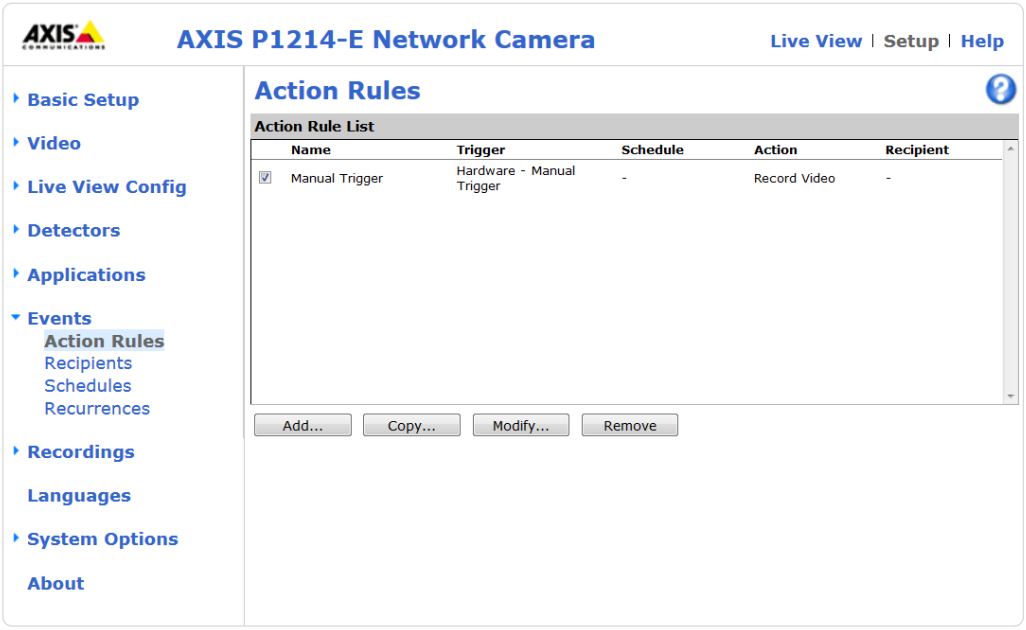
Kuva 7. PhoeniXin wlan-lähetin ja Axisin kamera

Axisin kamera ja kameran käyttöliittymä osoittautuivat loistavaksi kokonaisuudeksi. Kameran kuvan laatu on tarkka ja kameraan on mahdollista asettaa muistikortti, johon haettua kuvaa on käyttöliittymältä käsin helppo tallentaa. Lisäksi käyttöliittymä tarjoaa monipuolisia vaihtoehtoja kuvan tallentamista varten: Videokuvaa on mahdollista

tallentaa jatkuvasti, annetun signaalin perusteella tai ottaa yksittäisiä kuvia satunnaisin väliajoin. Lisäksi kameran käyttöliittymä tarjoaa mahdollisuuden tallentaa haettua videokuvaa joko paikalliselle muistikortille tai suoraa verkon yli serverille annettuun osoitteeseen. Tarjolla on myös mahdollisuus tallentaa kuvaa ensin kortille ja pilkkoa sitä näin pienemmiksi paketeiksi, jotka sitten kortilta siirretään serverille.

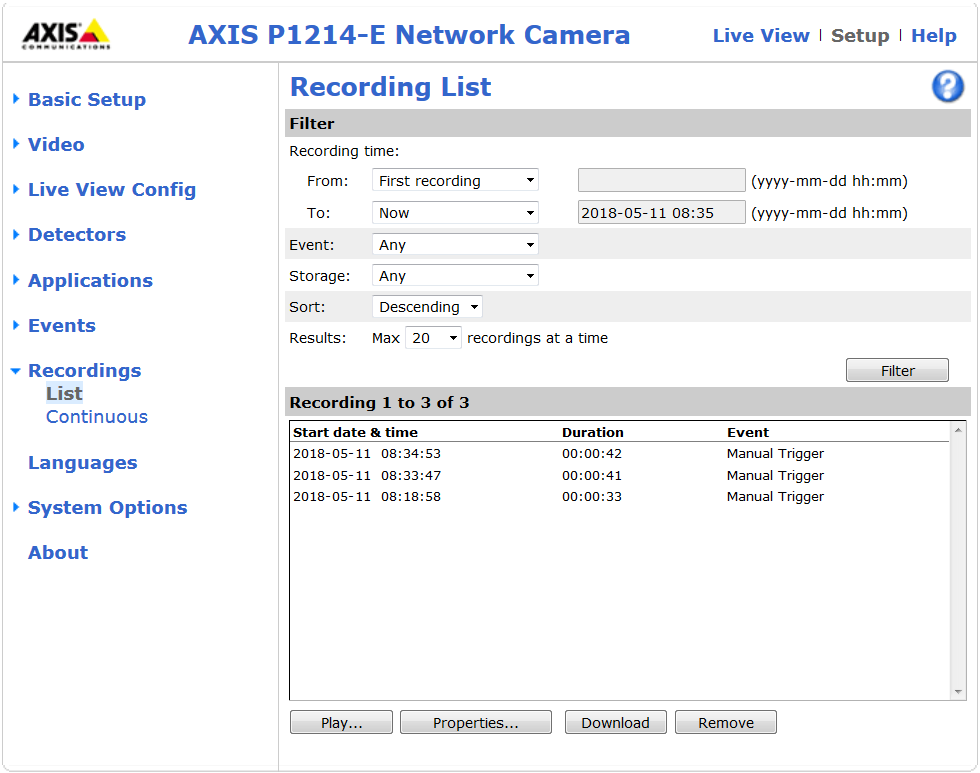
Koska kameran käyttöliittymä tukee haetun videokuvan tallentamista näin monipuolisesti, ei solun käyttöliittymälle kuvaa hakevan ja toistavan koodin tarvitsisi enää huolehtia kuvan tallentamisesta ja siirtämisestä serverille. Kameran käyttöliittymältä voidaan asettaa liipaisin-signaalin käyttö aktiiviseksi niin, että esimerkiksi robotin logiikalta saadun virhesignaalin ohjaamana videokuvaa aloitetaan tallentamaan, ja sen jälkeen lähettämään serverille annetuilla parametreilla, kuten halutun videokuvan tallennuksen aloitushetkellä ja lopetushetkellä.

Mikäli kameraan ei haluta kytkeä erikseen lähtöjä ja tuloja signaalia varten, voidaan liipaisimen anto hoitaa myös koodin avulla. Esimerkiksi Cimcorpin tilanteessa kameralla varustettavat robotit on kytketty varastonhallintajärjestelmään eli WCS:ään (Warehouse Control Software), joka saa tiedon myös robottien virheistä. Näin ollen WCS:n saadessa virheen robotilta voitaisiin tallentaminen aloittaa antamalla WCS:stä liipaisin-signaali palvelimelle. Tätä varten palvelimelle täytyy konfiguroida tiedot manuaalista liipaisinta varten (kuva 8.).



Kuva 8. Axis-kameran esimerkki-konfiguraatio manuaalisesta liipaisimesta

Kun manuaalinen liipaisin on konfiguroitu, voidaan se aktivoida suoraa kutsumalla sille varattua url-osoitetta. Osoitteen kutsuminen aktivoi liipaisimen, jonka jälkeen videotallenne tallentuu annettujen parametrien mukaisesti joko SD-kortille tai suoraan serverille (kuva 9.). Mikäli videoita halutaan ladata ja katsoa muilta käyttöpäätteiltä, on siihenkin mahdollisuus suoraa kameran käyttöliittymältä. Tämä edelleen helpottaa mm. tallenteiden saamista käyttöön esimerkiksi opetustarkoitukseen.



Kuva 9. Näkymä tallennetuista videotiedostoista

3.3 Kameroiden kiinnitys

Murata Machinerylta tilatuiden kameroiden pienen koon ansiosta mahdollisia kiinnityspaikkoja niille oli useita. Wlan-lähettimen kiinnityskohtakaan ei ollut varsinainen ongelma, sillä sen ei välttämättä tarvitse olla aivan kameroiden läheisyydessä, vaan kameroiden kiinnitysjohtoja voidaan jatkaa pidemmiksi näin mahdollistaen wlan-lähettimen kiinnityksen käytännössä mihin tahansa. Tätä työtä tehdessä saadun kokemuksen kautta todettiin, että yksi kamera per robotti ei ole riittävä siihen, että käyttäjällä olisi käyttöliittymällä selkeä näkymä robottia käsin ohjattaessa.

Jos esimerkiksi robotin ainoa kamera kiinnitettäisiin robotin pystyakselin suuntaista liikettä tekevään osaan, ei saadun kuvan perusteella olisi mahdollista nähdä robotin syvyyssuunnassa tekemää liikettä, eikä näin ollen olisi nähtävissä oikeaa tartuntakorkeutta esimerkiksi rengaspinoa nostaessa. Pystyakselilla kamera on kuitenkin tarpeellinen, sillä se kuvaa x- ja y-suunnissa tapahtuvaa liikettä nostettavan kohteen yläpuolella.

Jos kamera kiinnitettäisiin vaakaliikettä tekevään osaan välimatkan päähän pystyakselin suuntaista liikettä tekevästä robotin osasta, olisi syvyyssuuntainen liike hyvinkin havaittavissa kameralta saadusta kuvasta. Tällaisessa tilanteessa ei käyttäjällä yhden kameran järjestelmässä kuitenkaan olisi näkymää suoraa nostettavan kohteen yläpuolelta pystyakselilta katsottuna, joten robottia ei olisi mahdollista käsin ohjata hallitusti ja tarkasti nostokohteen yläpuolelle.

Kuva 10. Esimerkki kameroiden kiinnityspaikoista

Edellä mainittujen esimerkkien takia siis todettiin, että yksi kamera ei riitä takaamaan monipuolista näkymää käyttöliittymällä. Mutta mikäli kameroita olisi kaksi, joista toinen kiinnitettäisiin robotin pystyakselin suuntaista liikettä tekevään osaan ja toinen vaakaliikettä tekevään osaan pienen välimatkan päähän pystyakselin suuntaan liikkuvasta osasta, olisi kameroilta saatu näkymä monipuolinen ja riittävä, jotta käyttäjä voisi sen perusteella operoida robottia. (Kuva 10)

Mikäli kameroita olisi saatavilla kolme per robotti, voitaisiin kolmas kamera kiinnittää seuraamaan robotin ulossyöttöä. Tämän kameran tarkoituksena olisi tallentaa ja näyttää ulossyötössä mahdollisesti tapahtuvat virheet. (Kuva 11)

Kuva 11. Kamera laatikoiden ulossyötössä

3.4 Ohjelmisto

Solun käyttöliittymä on tehty Javalla, joten myös kamerakuvan hakemisen toteutus ja esittäminen solun käyttöliittymällä toteutettiin erillisenä Java-kokonaisuutena, joka on itsessään mahdollista upottaa lähes mihin tahansa Java-ympäristöön. Lopputuloksena

syntyi siis CcIpCam.jar-kirjasto. CcIpCam.jar:n luomisessa on käytetty avoimen lähdekoodin OpenCV-kirjastoa, joka sisältää metodeja kuvan hakemisesta, esittämistä ja muokkaamista varten.

CcIpCam.jar sisältää kaksi luokkaa: WebCamLoop.java ja IpCamJLabel.java. Näistä ensimmäinen hoitaa kuvan hakemisen, esittämisen ja päivittämisen JLabel-elementissä (kuva 12), joka taas luodaan IpCamJLabel.java:ssa.

```
66      @Override
67      public void run() {
68
69          try {
70              grabber.start();
71
72              while (running) {
73
74                  // Skip the image update if the camerascreen is not visible
75                  if (!this.ipCamJLabel.isVisible()) {
76                      continue;
77                  }
78
79                  IplImage grabbedImage = Java2DFrameUtils.toIplImage(grabber.grab());
80                  IplImage resizedImage;
81
82                  // If image couldn't be loaded, set error image to label
83                  if (grabbedImage == null) {
84                      throw new Exception("Null IPLImage!");
85                  }
86
87                  IplImage origImg = null;
88                  while (origImg == null || origImg.isNull() || origImg.width() == 0 || origImg.depth() == 0) {
89                      origImg = Java2DFrameUtils.toIplImage(grabber.grab());
90                  }
91
92                  //IF RECORDING, SAVE THE FRAME
93                  //                  if (currentlySavingVideo)
94                  //                      recorder.record(origImg);
95
96                  int width = 1020, height = 820;
97                  if (ipCamJLabel.getWidth() != 0) {
98                      width = ipCamJLabel.getWidth();
99                  }
00                  if (ipCamJLabel.getHeight() != 0) {
01                      height = ipCamJLabel.getHeight();
02                  }
03                  //              System.out.println("width=" + width + ", height=" + height + ", depth=" + origImg.depth()
04                  //                      + ", channels=" + origImg.nChannels());
05
06                  //DISPLAY THE IMAGE IN UI
07                  resizedImage = IplImage.create(width, height, origImg.depth(), origImg.nChannels());
08
09                  opencv_imgproc.cvResize(origImg, resizedImage);
10                  this.ipCamJLabel.setIcon(new ImageIcon(Java2DFrameUtils.toBufferedImage(resizedImage)));
11
12                  Thread.sleep(25);
13
14              }
15
16              grabber.stop();
17
18          } catch (Exception e) {
19              e.printStackTrace();
20              running = false;
21              this.ipCamJLabel.setErrorImage();
22          }
23      }
```

Kuva 12. Kuvan päivittäminen WebCamLoop.java:ssa

Rajapintoja luokka ei sisällä, vaan sen käyttö tapahtuu pelkästään konstruktoreja kutsumalla. Esimerkiksi Cimcorpin solun käyttöliittymään tehtiin uusi ikkuna (kuva 11), jossa on vaadittavat napit robottien ohjaukselle ja paneeli, jossa IpCamJLabelin konstruktoria kutsutaan parametreilla kameran lähetyksen ip-osoite, halutun videokuvan

leveys ja halutun videokuvan korkeus (kuva 13). Näin IpCamJLabel luo halutun kokoisen JLabel-elementin ja kutsuu WebCamLoop.javan konstruktoria, joka sitten esittää videokuvaa JLabel-elementissä.

```
/*
* CameraImageDisplayer
*/
private IpCamJLabel getCameraImageDisplayer() {
    if (ipCamJLabel == null) {

        ipCamJLabel = new IpCamJLabel("http://192.168.2.10:8080/?action=stream", cameraScreenWidth, cameraScreenHeigth);
    }
    return ipCamJLabel;
}
```

Kuva 10. Esimerkki IpCamJLabelin kutsumisesta

IpCamJLabel sisältää muitakin metodeja kuin ainoastaan WebCamLoop.java:n kutsumisen. Taulukossa 1. on listattu molempien luokkien käytettävissä olevat metodit ja kuvaukset niiden toiminnasta.

Taulukko 1. CcIpCam:in sisältämät metodit.

IpCamJLabel.java	WebCamLoop.java
setErrorImage(String filename) Asettaa JLabel –elementtiin virhekuvan, mikäli videokuvan haku epäonnistui.	**run()** Päivittää videokuvaa
setLoadingImage(String loadingImage) Asettaa JLabel –elementtiin latauskuvan siksi aikaa, kun videokuvaa haetaan	**startRecording()** Aloittaa videokuvan nauhoittamisen
start() Aloittaa videokuvan haun luomalla WebCamLoopin	**generateNewRecordingName(String prefix)** Generoi jokaiselle luodulle video-tiedostolle uniikin tiedostonimen
stop() Lopettaa videokuvan haun	**stopRecording()** Lopettaa videokuvan tallentamisen

CcIpCam.jar ei ole millään tapaa kamerasidonnainen eikä vaadi tiettyjen komponenttien käyttöä. Ohjelma hakee annetusta ip-osoitteesta sinne syötetyn videokuvan yksi ruutu (frame) kerrallaan oletuksena 20-kertaa sekunnissa huolimatta siitä, minkä kokoinen tai laatuinen kuva on tai millä laitteella se on sinne lähetetty. Toisin sanoen ainut vaatimus kuvanhaulle on se, että kuva on alun perin onnistuneesti lähetetty annettuun ip-osoitteeseen.

Kuva 13. Solun käyttöliittymän näkymä, jossa IpCamJLabelia on käytetty

3.4.1 Ohjelmiston ongelmat

CcIpCam.jar ei ole vielä valmis, vaan se vaatii edelleen jatkokehittämistä. Sen perusominaisuudet ovat toimivia, ja niitä voidaan käyttää tuotannossa, mutta kehitettävää kokonaisuudessa on vielä paljon. Esimerkiksi videokuvan tallennuksen kehitys on vasta alkumetreillä, eikä se ominaisuutena ole tällaisenaan vielä läheskään valmis tuotantoon.

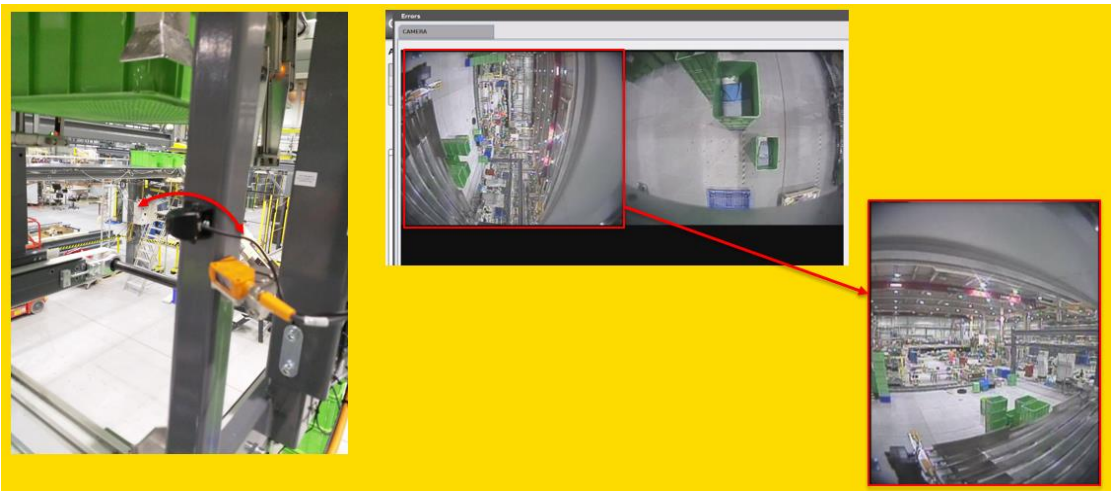
Videokuvan tallennus ohjelmiston avulla on tällä hetkellä mahdollista, mutta vain lokaalisti koodia ajavaan tietokoneeseen. Mikäli tallennusominaisuus haluttaisiin tuotantokäyttöön, tulisi sen mahdollistaa etätallennus verkon ylitse käyttäjän määrittämään kohteeseen. Lokaalisti tallentamisen ongelmana mm. Cimcorpin solu-tuotteessa on solutietokoneen muistin rajallinen määrä eikä videokuvan sinne tallentaminen tule kysymykseenkään.

Mikäli etätallennus haluttuun kohteeseen saataisiin tehtyä, tarvitsisi selvittää myös keino saada materiaali tallennetusta kohteesta järkevästi käyttöön. Jos kaikki videomateriaali päätyy toisaalle talteen, mutta sen noutaminen tai toistaminen sieltä on joko työlästä tai muuten haastavaa, ei siitä ole juurikaan hyötyä. Materiaalin pitäisi olla

helposti saatavilla ja käsiteltävissä, jotta sen tutkiminen, hallinnointi ja käyttäminen esimerkiksi koulutuskäytössä, olisi vaivatonta.

Lisäksi ohjelmistosta puuttuu vielä ominaisuus, jolla voitaisiin tallentaa ainoastaan halutut ajankohdat haetusta videokuvasta. Tällä hetkellä ohjelmisto tukee ainoastaan jatkuvaa tallentamista, joka ei ole järkevää mm. tilankäytön takia. Jatkuva tallentaminen aiheuttaa sen, että tallennettua materiaalia tulisi valtavasti ja näin tarpeellisen tallennetun materiaalin löytäminen ja parsiminen olisi haastavaa ja aikaa vievää. Ohjelmisto siis vaatisi jatkokehityksessä metodin, jonka avulla tallennettaisiin ainoastaan halutut tapahtumat.

Nykyisellään ohjelmistossa ei ole tukea haetun videokuvan orientaatiolle. Mikäli haettu videokuva tai osa kuvan kokonaisuudesta on lähteestä johtuen esimerkiksi 90-astetta vinossa, ei ohjelmiston avulla ole mahdollista kääntää kuvaa oikeaan asentoon. Tällainen ongelma voi ilmaantua, jos esimerkiksi kameran kiinnitys ei ole suorassa (kuva 14.). Ongelman aiheuttaa se, että nykyisessä muodossaan ohjelmisto hakee annettuun ip-osoitteeseen lähetetyn videokuvan kokonaisuudessaan, eikä esimerkiksi ensin erottele eri kameroiden lähettämiä kuvia ja myöhemmin liitä niitä yhdeksi kokonaisuudeksi, jotta niitä voitaisiin yksilöllisesti kääntää.



Kuva 14. Orientaatio-ongelma

Orientaatiosta päästään siihen, että ohjelmisto voisi tukea muitakin kuvankäsittelyyn liittyviä ominaisuuksia. Esimerkiksi zoomaus, yhden kamerakuvan valinta aktiiviseksi klikkaamalla tai kamerakuvaikkunan koon muuttaminen raahaamalla kuvan kulmasta

olisivat ominaisuuksia, joita käyttäjä saattaisi kaivata ohjelmistoon. Nämä eivät kuitenkaan ole kriittisiä normaalin operoinnin kannalta, joten niitä ei tässä työssä ole sen enempää käsitelty.

4 KEHITYS

Ohjelmiston tuottaminen ja liittäminen jo ennestään olemassa olevaan tuotteeseen asetti muutamia vaatimuksia ja rajoituksia siihen, minkälainen tulevan ohjelmiston tulisi olla. Jo ennestään olemassa olevaan ympäristöön suunnitellessa uutta ohjelmistoa on uuden ohjelmiston mukauduttava tähän ympäristöön niin ohjelmointikielensä kuin rakenteensa puolesta. Tässäkin tapauksessa ohjelmiston tuli olla mm. Javalla tehty JLabel-elementti, sen versionhallinta oli tehtävä Gitiä käyttäen, riippuvuuksien hallinta hoidettava Gradlella ja jatkuva järjestelmäintegraatio varmistettava Jenkinsillä.

4.1 Testausympäristö

Ohjelmistoa kehittäessä tarvittiin kehitettävälle ohjelmistolle testaus- ja kehitysympäristö, jossa kehitteillä olevaa koodia oli mahdollisuus päästä testaamaan käytännössä. Koska tavoitteena oli luoda periaatteessa mihin tahansa Java-ympäristöön uppoava kokonaisuus, ei myöskään testausympäristöä ollut rajattu muuhun kuin ohjelmointikieleen. Kuitenkin loogista oli käyttää testauksessa jo valmiina olevaa ympäristöä, johon tuleva ohjelmisto tultaisiin upottamaan. Näin ollen ohjelmistoa kehittäessä Cimcorpin solun käyttöliittymä tarjosi hyvän pohjan testata ohjelmiston toimivuutta, ominaisuuksia ja mahdollisia puutteita.

Pieniä muutoksia solun käyttöliittymä kuitenkin tarvitsi, ennen kuin sitä voitiin käyttää testauksessa. Ensinnäkin käyttöliittymälle oli luotava näkymä, jossa uutta Java-kokonaisuutta voitiin testata. Sen lisäksi oli käyttöliittymälle luotava toiminnallisuus, josta halutun testausympäristön näkymä saatiin avattua.

Valmiin tuotteen käyttöliittymän käyttäminen testauksen pohjana saattaa kuulostaa helpolta asialta toteuttaa, mutta käytännössä siihen liittyi myös monia haasteita. Solun käyttöliittymä on todella laaja paketti, joten sen toiminnan ja rakenteen sisäistäminen vei aikaa. Lisäksi muutoksien tekeminen valmiiseen tuotteeseen saattaa usein aiheuttaa monia arvaamattomia vaikutuksia ja näin ollen ohjelmoija saattaa luodessaan uutta ominaisuutta rikkoa toisen ominaisuuden toiminnan toisaalla.

Git oli haluttuun tavoitteeseen erinomainen versionhallintaohjelmisto. Solu-projektin päähaarasta irrotettiin oma kehityshaaransa eli branchinsa kamerakuvan hakua ja toistoa tukevalle versiolle, jota sitten voitiin kehittää päähaaran rinnalla häiritsemättä päähaaran kehitystä. Tämän ansiosta kehittäminen oli sen suhteen turvallista, että mikäli jotain hajoaisi, ei päähaara siitä häiriintyisi millään tapaa.

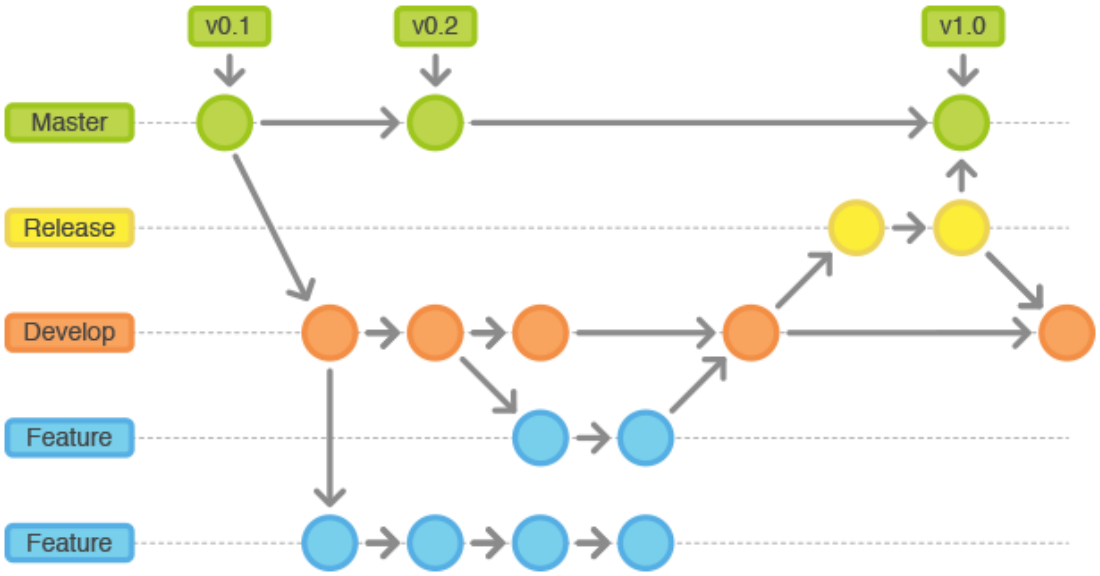
Muun muassa solu-projektin laajuudesta ja monipuolisuudesta johtuen solun käyttöliittymän muokkaaminen valmiiksi testiympäristöksi vei aikaa ja itse ohjelmiston ulkopuolista koodaustyötä tuli melko paljon. Vaati jonkun verran työtä, että kamerakuvaa varten oleva ikkuna saatiin tehtyä ja kaikki sen ympärillä oleva toiminnallisuus (mm. kyseisen paneelin avaava nappi) valmistui. Tämä työ kuitenkin kannatti, sillä oikean tuotteen käyttäminen ohjelmistopalasen testauksessa ja kehityksessä oli valtava etu. Mikäli testausvaiheessa olisi käytetty vain tekaistua ympäristöä oikean tulevan ohjelmiston käyttökohteen sijaan, olisi voinut jäädä monia asioita ottamatta huomioon. Nyt esimerkiksi ohjelmistolla haetun ja toistetun kuvan koon skaalaaminen oli helppo sovittaa sopimaan valmiiseen tuotteeseen sekä robotin ohjainpainikkeiden asettelu samaan ruutuun oli helppoa, kun sen teki jo testaus- ja kehitysvaiheessa.

Soluympäristön käyttäminen testauksessa palkitsi myös siinä kohtaa, kun valmiiksi todettua kokonaisuutta lähdettiin testaamaan halliin, jossa oli mahdollista nähdä solun ja uuden ohjelmiston toiminta oikeassa ympäristössään. Mikäli testaus olisi alun perin tehty vain tekaistulla testiympäristöllä, olisi solun käyttöliittymään joka tapauksessa tarvinnut uusi ohjelmisto upottaa. Nyt kun uuden ohjelmiston upotus oli tehty jo kehitysvaiheessa, ei sitä tarvinnut tehdä uudelleen.

4.2 Git

Git on versionhallintaohjelmisto, joka Cimcorpin tuotekehityksessä on yleisesti käytössä. Git on suunniteltu olemaan mahdollisimman nopea ja tehokas ja se tukee hajautettua työskentelyä sekä estää datan virheellisyyttä ja sen katoamista. Gitin yksi kiistattomista eduista on nimenomaan sen hajautetun työskentelyn tukeminen, mikä mahdollistaa monen eri ihmisen työskentelyn saman asian äärellä (kuva 15). Git on

julkaistu avoimen lähdekoodin lisenssillä, mikä takaa sen ilmaisen käytön ja jakamisen. (Git; GitHub Octoverse)



Kuva 15. Gitin kehityspuun haarautuminen

Gitin historia alkaa vuodesta 2005, jolloin Linus Torvalds aloitti kirjoittamaan omaa hajautettua versionhallintajärjestelmäänsä vanhan BitKeeper-ohjelmiston korvaajaksi. Torvaldsin vaatimukset uudelta versionhallintajärjestelmältä olivat seuraavat:

- Nopea
- Yksinkertainen design
- Vahva tuki ei-lineaariselle kehitykselle (tuhansia rinnakkaisia haaroja)
- Vapaa levitys
- Mahdollisuus käsitellä tehokkaasti suuria projekteja, kuten Linuxin ytimen projekti

Lopputuloksena tästä oli Git. Vuoden 2017 loppuun mennessä Gitillä oli yli 24-miljoonaa käyttäjää ja sitä on käytetty yli 67-miljoonassa projektissa. (Git; GitHub Octoverse)

Tämänkin ohjelmiston kehityksessä Gitistä on ollut suuri apu. Sen tuki ei-lineaariselle kehitykselle on mahdollistanut työskentelyn solu-projektin sivuhaarassa käyttäen sitä alun perin testiympäristönä ja näin ollen sallinut sen edelleen kehittämisen sisältäen videokamerakuvan hakemisen ja toistamisen sisältävän ominaisuuden. Myöhemmin, kun kyseinen ominaisuus todetaan toimivaksi ja tuotantokelpoiseksi, on Gitissä helppo

yhdistää uuden ominaisuuden haara alkuperäiseen kehityksen päähaaraan ja näin ottaa yleisesti uusi ominaisuus käyttöön.

4.3 Jenkins

Jenkins on avoimeen lähdekoodiin perustuva automaatio-serveri. Se toimii jatkuvan järjestelmäintegraation työkaluna ja mahdollistaa ohjelmiston koodin jatkuvan seurannan läpi sen kehittämisen.

Jenkins on mahdollista asettaa seuraamaan kaikkia mahdollisia koodimuutoksia projektissa, rakentamaan projektin koodeja automaattisesti käyttäen apunaan työkaluja kuten Ant ja Maven, ajamaan testejä ja niiden perusteella joko palauttamaan ohjelmisto takaisin viimeiseen toimivaan pisteeseen tai ilmoittamaan käyttäjälle mahdollisesta tapahtuneesta virheestä projektin koodien rakentamisessa (Kuva 16, Smart, J. 2011, 1-2).

Kuva 16. Projektin etusivu Jenkinsissä

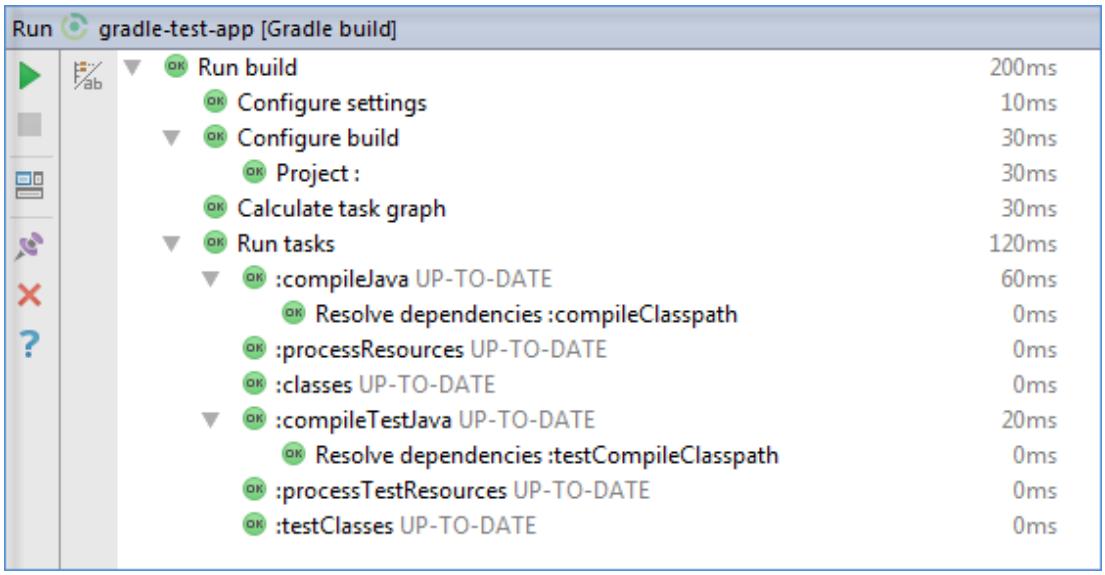
Jenkinsin julkaisi alun perin Kohsuke Kawaguchi vuonna 2004 nimellä Hudson työskennellessään Sunilla. Tuolloin Kawaguchi työsti Hudsonia vielä harrastuksenaan töiden ohessa. Vasta neljä vuotta myöhemmin vuonna 2008 Sun havaitsi Hudsonin potentiaalin ja pyysi Kawaguchia työskentelemään Hudsonin kanssa täysipäiväisesti. Kawaguchin ja muun kehitystiimin työpanoksen siivittämänä Hudsonista tuli jatkuvan järjestelmäintegraation markkinajohtaja 70 % markkinaosuudellaan vuoteen 2010 mennessä. (From Hudson to Jenkins)

Vuonna 2009 Oracle osti Sunin. Vuoden 2010 loppuun mennessä kiistat Hudsonin kehitystiimin ja Oraclen välillä aiheuttivat sen, että Oracle haki itselleen oikeudet nimeen Hudson. Erimielisyyksistä johtuen Hudsonin kehitystiimi siirsi Hudsonin pohjakoodit irralleen Oraclesta uudeksi projektiksi, joka nimettiin Jenkinsiksi ja jota Kawaguchi lähti seuraajiensa avulla kehittämään. Suurin osa Hudsonin käyttäjistä siirtyi hyvin pian Jenkinsin käyttäjiksi ja näin Jenkinsistä tuli suosituin jatkuvan integraation työkalu. (From Hudson to Jenkins)

4.4 Gradle

Gradle on avoimeen lähdekoodiin perustuva koodien rakennusautomaatiojärjestelmä. Se tukee monia ohjelmointikieliä kuten Javaa, Scalaa, Androidia, C/C++:ssaa ja Groovya ja monia tuhansia eri liitännäisiä. Yli neljän miljoonan kuukausittaisen latauskertansa ansiosta Gradle on suosituin rakennusautomaatiojärjestelmä. Esimerkiksi LinkedIn, Android, Netflix ja Adobe käyttävät apunaan Gradlea. (Gradle Build Tool)

Gradlea on erityisen suosittu suuren muokattavuutensa, nopeutensa ja tehokkuutensa ansiosta. Gradlea on helppo muokata mieleisekseen monien saatavilla olevien pluginien avulla. Gradlen nopeus perustuu mm. siihen, että se käyttää aiempien saamiensa testien tuloksia muiden testien suorittamiseen, eikä tästä johtuen vaadi jälleen koko ketjun suorittamista uudelleen alusta alkaen. Gradlen tehokkuus perustuu sen laajaan tukeen monille kielille ja teknologioille ja Android onkin sen valinnut viralliseksi rakennustyökalukseen. (Gradle User Manual)



Kuva 17. Onnistunut Gradle-rakennus

Tässä työssä Gradlea on käytetty sen kätevän riippuvuuksienhallintansa johdosta: Gradle voidaan konfiguroida lataamaan ja rakentamaan projektin kaikki riippuvuudet automaattisesti sekä suorittamaan haluttuja tehtäviä eli taskeja (kuva 17.). Automaattinen riippuvuuksien hallinnointi ja rakentaminen on erityisen tärkeää, kun riippuvuuksia alkaa olemaan useita ja niiden hallinta manuaalisesti on suuren lukumääränsä takia todella raskasta.

5 YHTEENVETO

Java on ohjelmointikielenä aina ollut itselleni mieleinen. Kun Cimcorp tarjosi Javalla tehtävää työtä videokamerateknologian parissa, oli yhdistelmä Javasta ja videokamerateknologista mielestäni erittäin mielenkiintoinen. Työn tekemisen kautta pääsin kokemaan ja oppimaan paljon uusista teknologioista ja työkaluista, joita nykypäivän ohjelmistokehityksessä voidaan apuna käyttää.

Työn käytännönläheisyys lisäsi mielenkiintoa työhön entisestään. Oli todella palkitsevaa saada ensimmäinen onnistunut videokuva siirrettyä käyttöliittymälle ja myöhemmin nähdä toistetun videokuvan hyöty käytännössä robottia käsiajolla ajaessa. Läpi koko ohjelmiston kehittämisen työn konkreettisuus oli vahvasti läsnä ja ohjelmistoon tehtävät pienetkin muutokset saivat aikaan suuria vaikutuksia lopputuloksessa.

Haastavinta työn alkuvaiheessa oli päättää, miten kuvan hakemista ja toistamista käyttöliittymällä lähtisin toteuttamaan. Mahdollisuuksia ja variaatioita työn tekemiselle olisi ollut useita ja dokumentaatiota erilaisista vaihtoehdoista oli saatavilla jonkin verran. Mielestäni oli hyvä, että Cimcorpin vaatimukset tuotekehityksen puolelta mm. käytettävistä työkaluista olivat olemassa, sillä ne rajasivat ohjelmiston tekemistä sopivasti, mutteivat kuitenkaan liikaa.

Jenkins, Git ja Gradle olivat itselleni täysin vieraita ennen työn aloittamista. Työtä tehdessäni tutustuminen kaikkiin näihin kolmeen oli kuitenkin välttämätöntä ja sitä kautta koin henkilökohtaisesti niistä jokaisesta saadun kiistattoman hyödyn ohjelmistokehityksessä. Henkilökohtaisesti ajattelen niiden olevan nykyaikaisia ohjelmointityökaluja, jotka ovat tänä päivänä kiinteä osa ohjelmistokehitystä. Tämän takia koin niiden oppimisen ja niiden parissa työskentelyn erittäin tärkeäksi ja hyväksi mahdollisuudeksi syventää oppejani alasta. Lisäksi kaikkiin edellä mainittuihin kolmeen tutustuminen ja oppiminen oli tehty suhteellisen helpoksi, sillä ne ovat vakinaistaneet asemansa ja suosionsa käyttäjien keskuudessa, joten ohjeita ja oppaita niistä oli saatavilla todella paljon.

Kaiken kaikkiaan videokamerakuvan hyödyntäminen robotin käsiajossa oli työnä mielestäni todella mielenkiintoinen ja toivonkin, että huolimatta nykyisestä työnkuvastani

Cimcorpilla Software-osastolla saan silti jatkossa käyttää aikaani tämän ohjelmiston kehittämiseen yhdessä tuotekehityksen kanssa yhä pidemmälle.

LÄHTEET

From Hudson to Jenkins, Viitattu 29.3.2019. Saatavissa: https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch01s04.html

Git, Viitattu 29.3.2018. Saatavissa: https://git-scm.com

GitHub Octoverse 2017, The State of the Octoverse, 29.3.2018. Saatavissa: https://octoverse.github.com/

Gradle Build Tool, Accelerate developer productivity, Viitattu 29.3.2018. Saatavissa: https://gradle.org/

Gradle User Manual, Viitattu 29.3.2018. Saatavissa: https://docs.gradle.org/current/userguide/userguide.html?_ga=2.211755221.154819765.1524038697-696634940.1519821845

Smart, J. 2011. Jenkins: The Definitive Guide. Sebastopol: O'Reilly Media Inc.